Is Silence Golden?

Adi Yoaz
Ronny Ronen
Robert Chappell
Yoav Almog

Intel Corporation

intel®

Silent Stores

- A *silent store* is a write to memory that does not change the value at the target address
- Lepak and Lipasti reported this phenomenon at ISCA-27 and thoroughly investigated the causes
- 20% to 60% of stores are silent (29% on average) across winstone98, sysmark98, and SPECint2000

Benefits of Silent Stores

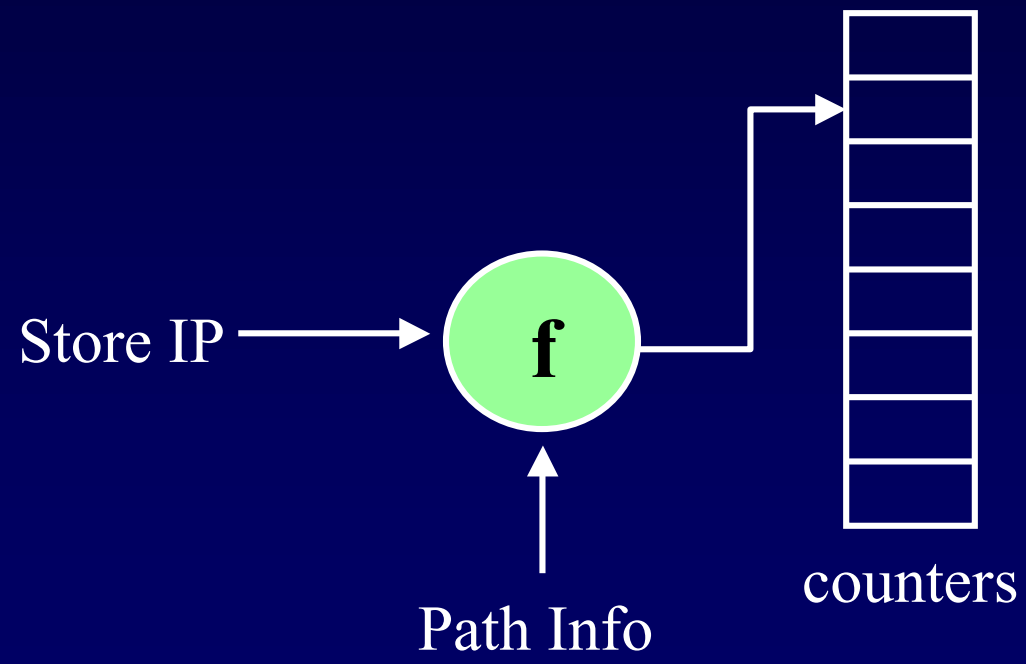
- So far (Lepak and Lipasti, ISCA-27, MICRO-33):
  - Save memory bandwidth
  - Potential for narrower interfaces
- These benefits are achieved by detection of silent stores at execution time
- Could silence be even more useful if predicted in the front end?
  - First, can we predict silent stores accurately?
  - If so, what can we do with the speculative knowledge?

Our Contributions

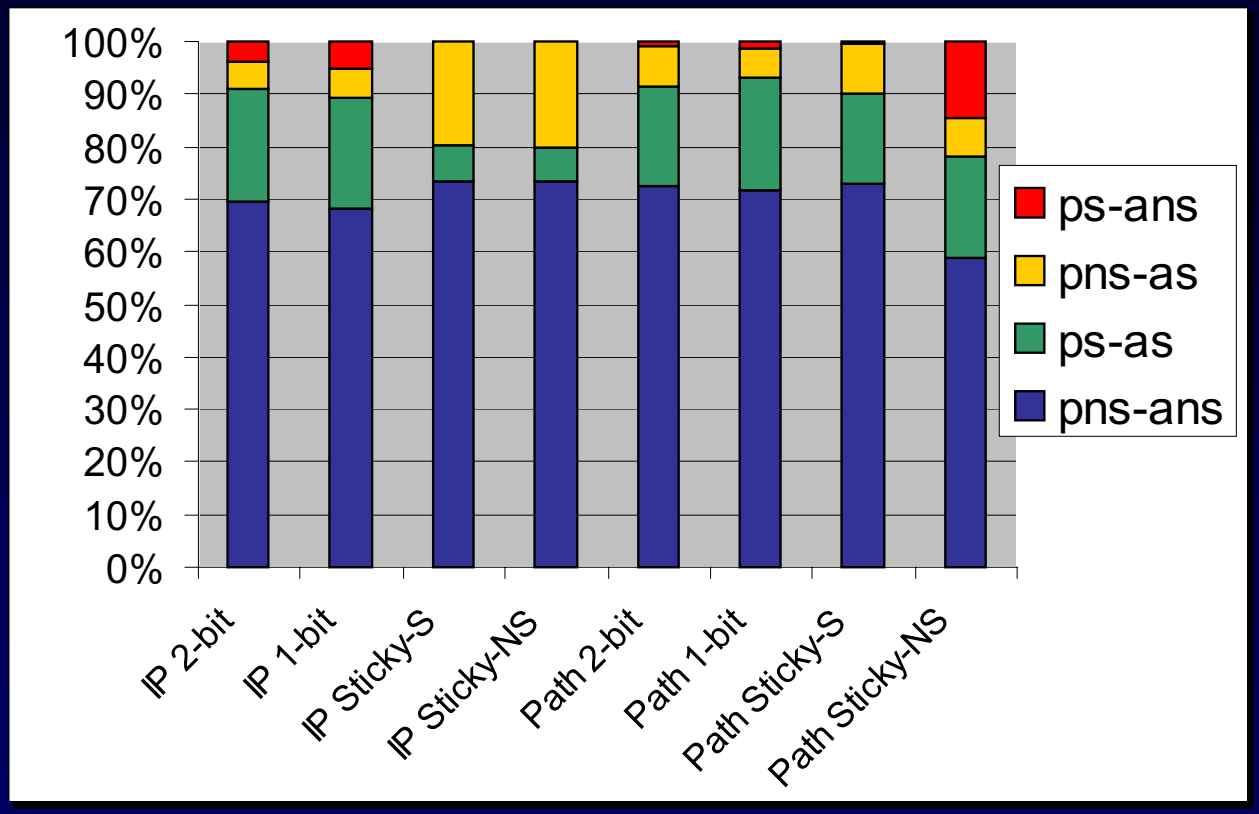
- Silent stores are highly predictable
  - We can build an accurate silent store predictor
- Silent stores become more useful with speculation
  - We can improve memory disambiguation
  - We can better exploit dead instructions

Building a Silent Store Predictor

- Leverage well-understood branch prediction technology



Predictor Accuracy

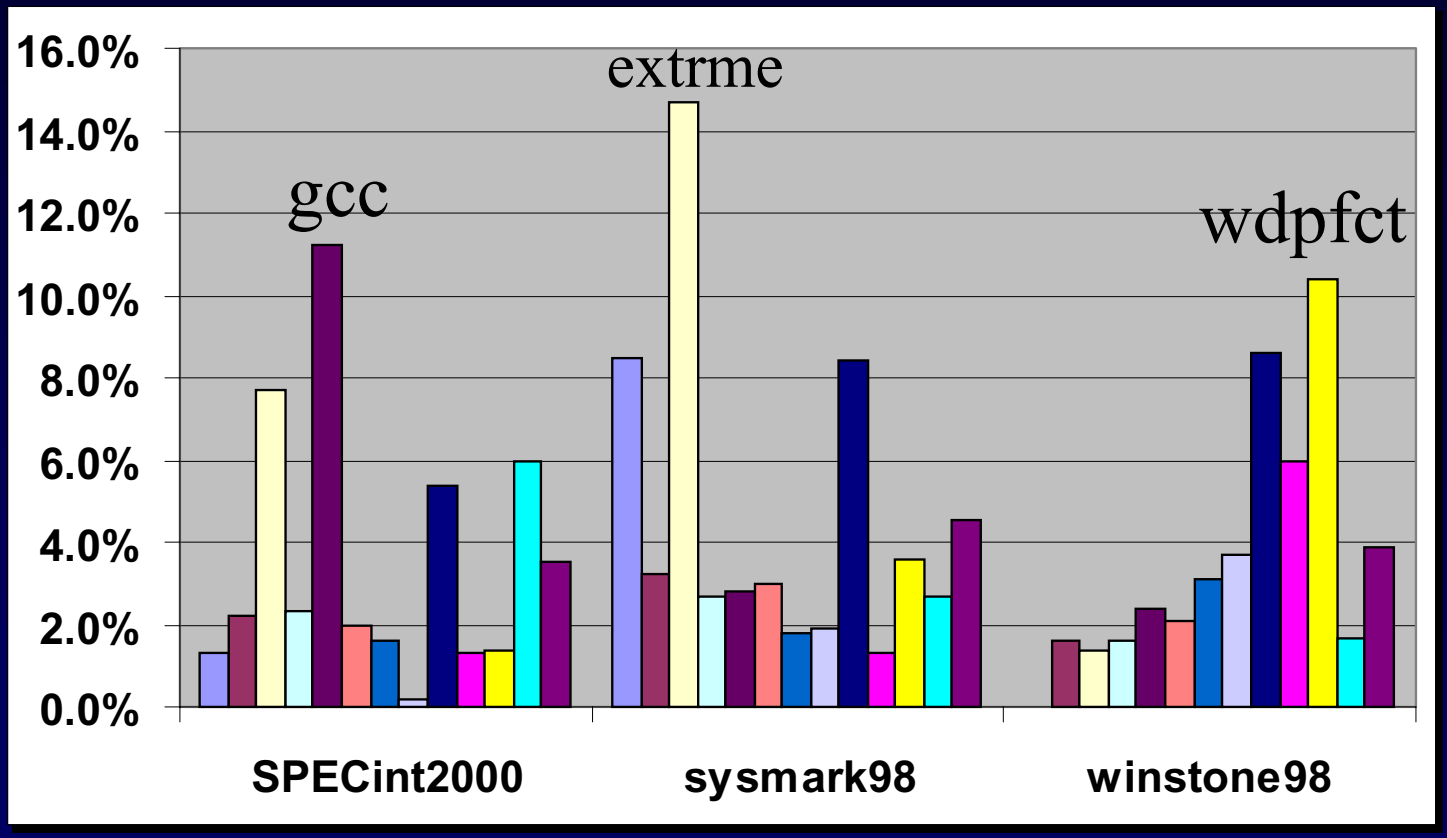


Path-based indexing and 2-bit counters yields best combination

Improving Memory Disambiguation

- Current dependency predictors try not allow loads to bypass colliding stores
- Loads can always bypass *silent* stores, even if the addresses collide
- We can improve dependency predictors by incorporating our silent store predictors
  - Can do better than current "perfect" dependency prediction

Performance Improvement Using Silent Store Predictions for Disambig



Means: SPEC 3.55%, sysmark 4.55%, winstone 3.87%

Better Exploiting Dead Instructions

- Found 1.7% of dynamic instructions are dead
  - Eliminating these instructions leads to little gain
- Silent stores are another type of “dead” instruction
  - Brings total to over 5%, increasing the potential benefits of exploiting dead instructions
- Address/data calculations leading to silent stores might also be dead
  - Further increases the total dead instructions

intel®

How to Better Exploit Dead Instructions

- Squash “naturally dead” instructions
- De-prioritize silent stores and related computations for performance and power
  - Keep history of computations leading to silent stores
  - Use silent store predictor to identify these cases
- Prioritization becomes more useful as aggressive clock speeds push towards more contention

Conclusions and Future Directions

- Speculating on silent stores yields increased advantages
  - Can help disambiguation and dependence prediction
  - Can help establish instruction priority for power and performance
- Continuing to:
  - Quantify effects
  - Understand how to best exploit dead instructions
  - Look for new types of dead instructions